Journal of Computing and Information Technology - CIT 14, 2006, 4, 329–336
doi:10.2498/cit.2006.04.09

An Approach to Grid Scheduling by Using Condor-G Matchmaking Mechanism

Emir Imamagic, Branimir Radic, Dobrisa Dobrenic

University Computing Centre, University of Zagreb, Croatia

Grid is a distributed environment that integrates computing, storage and other resources in order to enable execution of applications that cannot be run on a single resource. Such environment requires advanced scheduling system in order to efficiently execute users' applications.

In this paper, we give an overview of issues related to grid scheduling. We describe in details one of the most mature solutions – Condor-G Matchmaking mechanism. Furthermore, we propose our own approach to building grid scheduling system based on Condor-G Matchmaking.

Keywords: grid scheduling, execution management, Condor-G, matchmaking.

1. Introduction

Grid is a geographically distributed system that enables integration of heterogeneous resources needed for execution of complex, demanding scientific and engineering applications. Integration of resources beyond boundaries of a single organization also enables easier cooperation among diverse scientists who work in different geographical locations.

Grid middleware (GMW) is a set of services and protocols that enable seamless integration of resources in grid. It provides a layer of abstraction that hides differences in underlying technologies (e.g. computer clusters, storage managers, application services, etc.). Numerous standards are being defined for grid protocols and services, majority of them within Global Grid Forum (GGF) [5] organization. Basic functionalities of grid middleware are security, information, job and data management. Most widely used solutions that provide these basic functionalities are Globus Toolkit [10] and UNICORE [24].

Grid scheduling (also called superscheduling, metascheduling and grid brokering) is one of the advanced features of grid middleware. It is defined as the process of scheduling jobs where resources are distributed over multiple administrative domains [21]. Up to date, there is no grid scheduling system that fully meets the requirements that will be described in following sections.

Execution of jobs in grid is shown in Figure 1. It consists of the following activities:

- users submit their jobs described by using a description language to the grid scheduler (1),

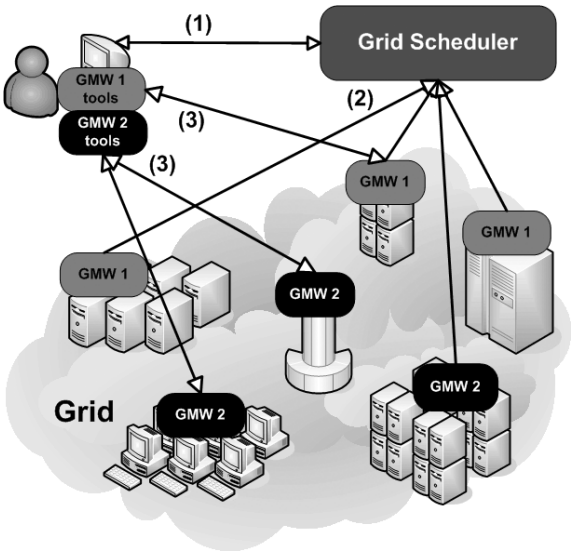

Fig. 1. Grid scheduling architecture.

- scheduler uses grid middlewares' information systems to discover and evaluate resources (2),
- once the scheduler has defined where the job will be executed, execution is started and managed by using available grid middleware components (e.g. job and data management systems) (3).

Grid scheduling differentiates from classical cluster batch scheduling in many ways. In case of grid, a scheduler does not have full control over resources, information about resources is usually unreliable and stale, application behavior is difficult to predict. All these reasons make grid scheduling far more difficult to realize.

This paper is organized as follows. After short introduction, in Section 2, we describe main issues related to grid scheduling. In Section 3, we describe how the Condor-G Matchmaking can be used for grid scheduling. We describe our grid environment in Section 4. In Section 5, we describe our proposed systems in detail. Related work is described in Section 6. Conclusions and future directions are given in Sections 7 and 8.

2. Grid Scheduling Issues

Due to the complexity of grid environment, grid scheduling raises numerous issues. In this section, we shortly describe the most important issues that scheduling system should meet.

Today, there are many *grid middlewares* that provide basic set of functionalities. Some of them are implemented in accordance with grid standards (e.g. Globus Toolkit 4.0) and some have become de facto standards (e.g. UNICORE and Globus Toolkit 2.4). Grid scheduling system should be able to utilize as many grid middlewares as possible. Here are some existing solutions for particular subsystems:

- Security management: Globus Grid Security Infrastructure (GSI) ([14]), UNICORE Security, Virtual Organization Membership Service (VOMS) ([25])
- Information systems: Globus Monitoring and Discovery System (MDS) 2 and MDS 4 ([11]), Ganglia ([5]), Network Weather Service ([19])
- Job management: Globus Grid Resource Allocation and Management (GRAM) 2 and GRAM 4 ([12]), UNICORE, Condor
- Data management: Globus GridFTP and Reliable File Transfer (RFT) ([13]), UNICORE File Management.

Ability to *refresh proxy certificates* in environments that utilize Globus Toolkit as grid middleware is preferred. The most widely used solution to this problem is using MyProxy credential repository ([21]).

Grid scheduler should support various *job types*. Basic types are serial and parallel jobs. Serial job is an application that demands single processor for execution. Parallel job requires more than one processors for execution (typically MPI applications). Some of the complex job types are job arrays and workflows. Job array comprises multiple execution of the same job (usually initialized with different parameters) and the workflow is a set of dependent tasks.

Standard and advanced functionalities of job management system should be supported:

- *Checkpointing* – ability to store job's current state, which can be used afterwards to reconstruct the job.
- *Preemption* – ability to evict one job from one resource, in favor of another with higher priority.
- *Job migration* – ability to dynamically move active job from one resource to another.
- *Rescheduling* jobs on alternative resources. This ability is important in cases when job does not execute properly on specific resources.
- *Fault tolerance* – ability to automatically recover from job or resource failure.
- *Advance reservation* of resources for a specific period of time.

A scheduler should be capable of assigning *priorities* to jobs and resources. In addition, resource owners should be able to define which jobs they prefer. In the same way, users should be able to rank the resources and request specific features (e.g. libraries or hardware capabilities).

Users should be able to define *custom scheduling algorithms*. Besides support for custom algorithm development, most common algorithms implementations should be provided.

Scheduling system should make use of local job management system's *advance reservations* mechanism. By using advance reservations, grid execution system enables users to run their applications in explicitly defined timeframe. In addition, by using advance reservations, parallel applications distributed over multiple clusters can be guaranteed synchronous startup of processes on all clusters.

A scheduler should take into account data location and movement (*dataaware scheduling*) and characteristics of network links. An example of data-aware scheduling is assigning jobs to resources closer to the data instead of moving large data over network.

System *scalability* is very important due to the nature of grid system. For example, grid scheduling system should be capable of handling continuous and massive load.

Detailed analysis of existing grid scheduling solutions by using criteria described above is presented in [18]. The conclusion is that systems GridWay ([16]) and CondorG satisfy most of the features. In the remaining part of this article, we will concentrate solely on Condor-G.

3. Condor

Condor [1] is a distributed environment developed at the University of Wisconsin, designed for high throughput computing (HTC) and CPU harvesting. CPU harvesting is a process of exploiting nondedicated computers (e.g. desktop computers) when they are not used.

Condor architecture is shown in Figure 2. Heart of the system is Condor Matchmaker. Users describe their applications with ClassAds language and submit them to Matchmaker. ClassAds allows users to define custom attributes for resources and jobs. On the other side, resources publish information to Matchmaker. Condor Matchmaker then matches job requests with available resources.

Condor provides numerous advanced functionalities described in Section 2, such as job arrays and workflows support, checkpointing, job migration, rescheduling, fault recovery. It enables users to define resource requirements and rank resources and mechanism for transferring files to/from remote machines.

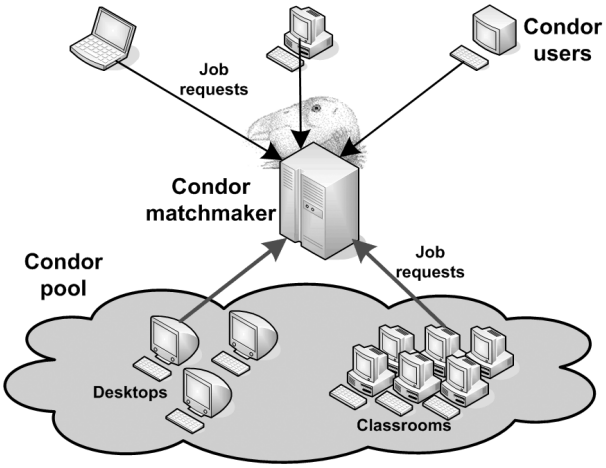

Fig. 2. Condor architecture.

Although utilized for integration of resources, Condor is intended to be used in smaller scale environments and should be seen more as a local job management system than a grid middleware.

3.1. Condor-G

Condor-G [5] is the Condor extension that enables using Condor tools for submitting jobs to grid. Currently supported grid middlewares are: Globus Toolkit versions 2.4, 3 and 4, UNICORE and NorduGrid. Additionally, Condor-G can use MyProxy server for storing user credentials in case of long running jobs.

However, Condor-G cannot schedule jobs, it simply executes the job on remote resource by using grid middleware. Condor-G Matchmaking mechanism extends Condor-G with the ability to use matchmaking algorithm to schedule jobs to grid.

Condor-G Matchmaking inherits almost all advantages of standard Condor system and Condor-G. It enables job scheduling based on complex job requirements. For example, a user can easily define specific requirements, such as storage space required, required libraries, etc. In addition, a user can prefer specific set of resources to others, by giving them a higher rank. Furthermore, using multiple Condor-G Matchmakers are supported. This feature enables distribution of workload, which is important for achieving greater scalability of the system.

One of the disadvantages of Condor-G Matchmaking is lack of support for parallel jobs. In current version users can submit parallel jobs, however Condor-G Matchmaker will not be aware of jobs parallelism. It will simply assume that it is dealing with serial jobs. Furthermore, there is no implicit data-aware scheduling, although users can explicitly define resources, closer to input data are preferred. Also, in current version, it is not possible to define custom scheduling algorithm.

Another issue is that Condor-G Matchmaking lacks integration with grid information systems. In order to use Condor-G Matchmaking, one has to develop a custom system that will provide information about resources to Condor-G Matchmaker.

4. CRO-GRID Grid Environment

CRO-GRID [2] is the Croatian national grid initiative that consists of three projects and aims to build computational grid for science and enterprise needs. CRO-GRID Infrastructure project [3] is focused on building and maintaining grid infrastructure. CRO-GRID Mediator is developing serviceoriented grid middleware components based on recent WSRF specification. Four scientific applications are being developed for this grid within CRO-GRID Application project.

At this moment, CRO-GRID Infrastructure consists of five clusters placed in institutes in four cities. Local job management systems (JMS) used on clusters are Sun Grid Engine and Torque with Maui scheduler. The network used for CRO-GRID infrastructure is shown in Figure 3. In cooperation with Giga CARNet project ([7]), all links (besides the link to ETFOS cluster) have gigabit bandwidth.

Currently, we are using three grid middlewares: Globus Toolkit versions 2.4 and 4.0 and UNICORE. General idea was to deploy all three of them in order to gain practical experience and allow developers of applications to decide which one provides the best functionalities. Besides basic grid middleware, we deployed numerous additional services, such as GridSphere grid portal ([15]), MyProxy credential storage, Nagios monitoring system ([19]), etc.

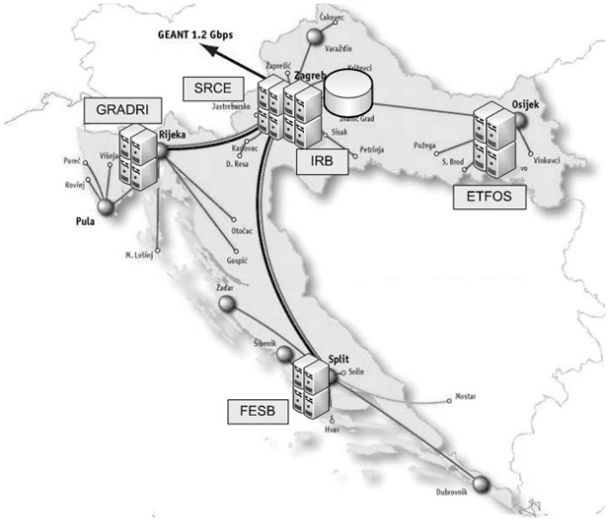

Fig. 3. CRO-GRID Infrastructure network.

At the same time, we want to provide users with the unique interface for submitting jobs to all three middlewares. By providing such interface, we could transparently choose between grid middlewares. In addition, such interface should enable resource owners to decide which grid middleware to use. It should also enable easy integration of resources from other grids.

5. CRO-GRID Condor-G Matchmaking Approach

After thorough investigation of existing grid scheduling systems ([18]) we decided to utilize Condor-G Matchmaking mechanism. Our rationale is the following. First, it meets most of the requirements described in Section 2. The most important requirement for us is the support for multiple grid middlewares. Second, it creates unique interface described in Section 4.

The basic idea is to utilize grid information systems (GIS) that are part of deployed grid middlewares. The information about available computational resources is retrieved from GIS and published to Condor-G Matchmaker. Once the resource state is changed, Publisher updates the Condor-G Matchmaker.

All this is completely transparent for the user. The user submits the job as it would be submitted to a standard Condor system. Once the match is made, Condor-G submits the job by using appropriate grid middleware.

The proposed system is currently in operational testing stage. Once the performance testing phase is finished, the system will be offered for use in CRO-GRID grid infrastructure. Details about our components and current state of development are described in the following subsections.

5.1. Proposed Architecture

The architecture of proposed scheduling system based on Condor-G Matchmaking is presented in Figures 4 and 5. The first figure presents central part of system and the second depicts the system from the user's aspect.

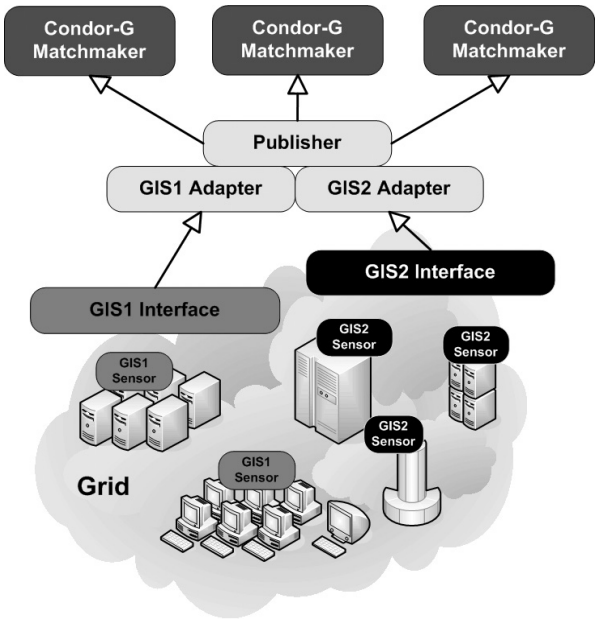

Fig. 4. CRO-GRID Condor-G Matchmaking (central part).

Central side of our system consists of three components: GIS Adapter, Publisher and CondorG Matchmakers. Publisher is responsible for providing information to Condor-G Matchmakers. GIS adapters enable Publisher to pull information from a particular type of GIS. We utilize Condor-G Matchmaking ability to use multiple Condor-G Matchmakers in order to accomplish greater scalability.

The user's perspective is shown in Figure 5. The user describes the job with ClassAds as in standard Condor system and uses Condor tools to submit it (1). Condor-G matchmaker assigns

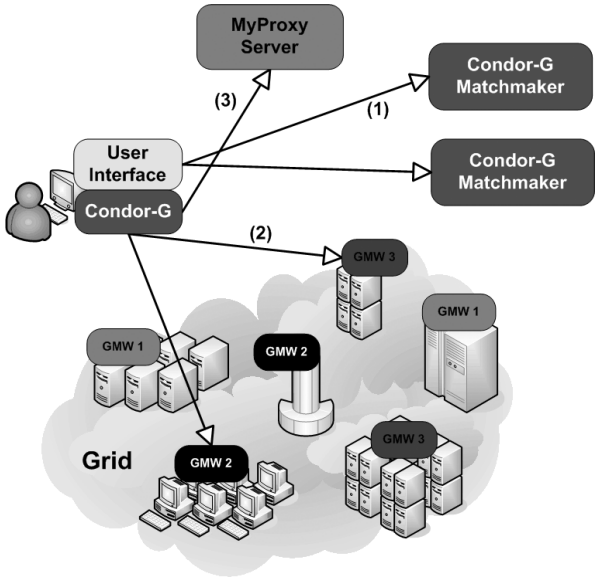

Fig. 5. CRO-GRID Condor-G Matchmaking (the user's perspective).

a set of grid resources to the job for execution. Condor-G utilizes appropriate underlying grid middleware components to execute the job (2). If needed, Condor-G enables refreshing of user's certificate by using MyProxy server (3). Our User interface component enables the user to retrieve additional, grid-specific information about the job.

5.2. Publisher

Publisher component is responsible for publishing information gathered from available GIS to Condor-G Matchmaker.

Special issue with Publisher is timing. The period between two publishing events (publish period) has to be tightly correlated with Condor-G Matchmaking scheduling period and GIS refresh period. Publish period has to be smaller than scheduling period, otherwise Condor-G Matchmaker will use stale information. On the other hand, publish period has to be longer than GIS period, otherwise stale information will be published.

As shown in Figure 4, Publisher is deployed independently of both GIS and Condor-G Matchmaker. Service itself does not require significant resources. Resource information can be published to multiple Condor-G Matchmakers. If that is the case, Publisher usually deploys close to the GIS interface (e.g. on the same server as GIS central service).

5.3. GIS Adapter

GIS Adapter is the component responsible for interpreting data from particular type of GIS. Publisher uses it for retrieving information from GIS.

Complexity of GIS Adapter depends on the information system (e.g. type of the API it provides, type of the format it uses, etc.). Currently we support Globus MDS2 and MDS4. In case of MDS2, we developed an adapter that can parse data in LDAP form. In case of MDS4, data is stored in XML form.

New GIS Adapters for additional information systems can be easily developed. By providing new Adapters, one can add support for other grid middlewares.

5.4. User Interface

Since the Condor tools do not provide extensive information about Condor-G Matchmaking, we developed wrappers around Condor tools. Client interface enables the user to see where the job is executed, which grid middleware is used and to get the job's status.

5.5. Advantages

By relying on GIS, we eliminate the need for the following components: resource adapters, sensor management and information management. We do not have to develop adapters for gathering information from each resource type (e.g. various local job management systems: Sun Grid Engine, Condor, etc). Next, we do not need to implement mechanisms and protocols for controlling sensors that collect information on individual resources. Last, but not least, grid information systems have to contain subsystems for management of the gathered information. Also, by eliminating these components we avoid introducing additional load to the resources.

As all grid middlewares need to have some sort of information system, we find that our approach will always be applicable.

5.6. Drawbacks

Existing GIS solutions such as Globus MDS2, MDS4 or R-GMA usually do not provide rich set of information about particular resources. Information is often limited to the set that each type of resource can provide. However, unreliability and lack of information are basic characteristics of grid systems.

6. Related Work

Because of its many advantages, Condor-G and Condor-G Matchmaking mechanisms are used by many grid projects.

EGEE's gLite ([8]) and its preceding project European DataGrid Project (EDG) ([4]) utilize Condor-G for submitting jobs in the Workload Management System (WMS). In this case, entire complex system is built around Condor-G. Therefore, the usage of WMS requires installation of additional gLite's components. Main difference is that our approach is much more lightweight and can be deployed by using existing grid technologies: Condor, Globus Toolkit, UNICORE and MyProxy.

Condor-G Matchmaking is used by many grid projects such as Grid X1 ([15]) and SAMGrid ([22]). Major difference between these approaches is that in their case information is gathered directly from the resources instead from GIS. In subsection 5.5 we described negative aspects of such approach.

7. Conclusion

Seamless job execution management is extremely important capability of computational grid. As we see it, today's systems are not mature enough for end users to use them extensively. Another issue of existing systems is limited grid middleware support. In order for scheduling system to be widely adopted, it should support as many grid middlewares as possible.

In this article, we propose scheduling system based on one of the most mature solutions – Condor-G Matchmaking mechanism. Main features of our approach are: full utilization of all

Condor-G functionalities, ability to extract information from diverse sources and ability to transparently switch between grid middlewares.

8. Future Work

We are planning to extend Publisher with the ability to utilize multiple grid information systems to retrieve different type of information about the same resource. By combining information, Publisher will be able to provide more accurate and detailed information about resources.

In future, we are also planning to extend our system in the following directions:

- Integrating proposed system into GridSphere grid portal environment in order to provide user-friendly interface.
- Extending the set of resource attributes published to Condor-G Matchmaker.
- Extending the set of supported grid information systems.
- Investigating approaches for managing parallel jobs.
- Integrating data scheduling service Stork with existing solution.

9. Acknowledgements

All given results were achieved through the work on the TEST program – Technological research-development projects with the support of the Ministry of science, education and sports.

References

[1] The Condor Project Homepage, http://www.cs.wisc.edu/condor [02/28/2006].

[2] CRO-GRID Homepage, www.cro-grid.hr [02/28/2006].

[3] CRO-GRID Infrastructure Project, www.srce.hr/crogrid/infrastructure/ [02/28/2006].

[4] The DataGrid Project, http://eu-datagrid.web.cern.ch/eu-datagrid/ [02/28/2006].

[5] J. FREY, T. TANNENBAUM, M. LIVNY, F. FOSTER, S. TUECKE, Condor-G: A Computation Management Agent for Multi-Institutional Grids, *Cluster Computing* 2002, 5(3), pp. 237–246.

[6] Ganglia Monitoring System, http://ganglia.sourceforge.net/ [02/28/2006].

[7] Giga CARNet, http://giga.carnet.hr/ [02/28/2006].

[8] gLite, Lightweight Middleware for Grid Computing, http://glite.web.cern.ch/glite/ [02/28/2006].

[9] Global Grid Forum, http://www.ggf.org [02/28/2006].

[10] Globus Alliance, http://www.globus.org [02/28/2006].

[11] Globus Toolkit Information Services, http://www.globus.org/toolkit/mds/ [02/28/2006].

[12] Globus Toolkit Execution Management, http://www.globus.org/toolkit/docs/4.0/execution/ [02/28/2006].

[13] Globus Toolkit Data Management, http://www.globus.org/toolkit/docs/4.0/data/ [02/28/2006].

[14] Grid Security Infrastructure, http://www.globus.org/Security/ [02/28/2006].

[15] GridSphere Project, http://www.gridsphere.org [02/28/2006].

[16] GridWay, http://www.gridway.org [02/28/2006].

[17] Grid X1, A Canadian Computational Grid, http://www.gridx1.ca/ [02/28/2006].

[18] E. IMAMAGIC, B. RADIC, D. DOBRENIC, CRO-GRID Grid Execution Management System. In: *Proceedings of the 27th International Conference on Information Technology Interfaces*, 2005 Jun 20-23; Cavtat, Croatia. Zagreb: SRCE University Computing Centre, 2005. pp. 77–83.

[19] Nagios, http://www.nagios.org/ [02/28/2006].

[20] Network Weather Service, http://nws.cs.ucsb.edu/ [02/28/2006].

[21] J. NOVOTNY, S. TUECKE, V. WELCH, An Online Credential Repository for the Grid: MyProxy. In: *IEEE International Symposium on High Performance Distributed Computing*; 2001 Aug 7-9; San Francisco, USA. pp. 104–114.

[22] SAMGrid http://www-d0.fnal.gov/computing/grid/ [02/28/2006].

[23] J. M. SCHOPF, General Architecture for Scheduling on the Grid; 2002, http://www-unix.mcs.anl.gov/~schopf/Pubs/sched.arch.2002.pdf [02/28/2006].

[24] UNICORE, http://unicore.sourceforge.net/ [02/28/2006].

[25] Virtual Organization Membership Service (VOMS), http://infnforge.cnaf.infn.it/voms/ [02/28/2006].

Received: June, 2006
Accepted: September, 2006

Contact addresses:
Emir Imamagic
University Computing Centre
University of Zagreb
Croatia
emir.imamagic@srce.hr

Contact addresses:
Branimir Radic
University Computing Centre
University of Zagreb
Croatia
branimir.radic@srce.hr

Contact addresses:
Dobrisa Dobrenic
University Computing Centre
University of Zagreb
Croatia
dobrisa.dobrenic@srce.hr

EMIR IMAMAGIC graduated from the Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, in May 2004. His research interests are high performance computing, distributed computing, computer clusters and grid systems.

Before graduation, he worked on the AliEn Grid project at CERN, Switzerland in the summer of 2003 and on the MidArc middleware project at Ericsson Nikola Tesla in the summer of 2002. He is currently working as a researcher on the CRO-GRID Infrastructure project and the EGEE-II project at the University Computing Centre.

BRANIMIR RADIC graduated from the Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, in July 2004. his research interests are high performance computing, distributed computing, computer clusters and grid systems.

Before graduation, he worked on the CRistal project at CERN, Switzerland in the summer of 2003 and on the MidArc middleware project at Ericsson Nikola Tesla in the summer of 2002. He is currently working as a system administrator, researcher on the CRO-GRID Infrastructure project and researcher on the EGEE-II project at the University Computing Centre.

DOBRISA DOBRENIC graduated from the Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, in 1991.

He has worked at the University Computing Centre (SRCE) since 1992, heading the Department of Computer Systems since 1994. His responsibilities include designing, building and managing computer structures in the Croatian academic and research community, including security systems, as well as various expert referral centres, helpdesks and education for other specialists.

Currently, he is the head researcher in the national project CRO-GRID Infrastructure, and the national manager of all grid services in EU FP6 project EGEE-II.